平成19年度　春期
基本情報技術者
午後　問題

試験時間	13:00 ～ 15:30（2時間30分）

注意事項

1. 試験開始の合図があるまで，問題冊子を開いて中を見てはいけません。
2. この注意事項は，問題冊子の**裏表紙**に続きます。必ず読んでください。
3. **答案用紙への受験番号などの記入は，試験開始の合図があってから始めてください。**
4. 問題は，次の表に従って解答してください。

問題番号	問1 ～ 問5	問6 ～ 問9	問10 ～ 問13
選択方法	全問必須	1問選択	1問選択

5. 答案用紙の記入に当たっては，次の指示に従ってください。
   (1) HB の黒鉛筆又はシャープペンシルを使用してください。訂正の場合は，あとが残らないように消しゴムできれいに消し，消しくずを残さないでください。
   (2) 答案用紙は光学式読取り装置で処理しますので，答案用紙の**マークの記入方法**のとおりマークしてください。
   (3) **受験番号欄**に，**受験番号**を記入及びマークしてください。正しくマークされていない場合，答案用紙の**マークの記入方法**のとおりマークされていない場合は，採点されません。
   (4) **生年月日欄**に，受験票に印字されているとおりの**生年月日**を記入及びマークしてください。正しくマークされていない場合は，採点されないことがあります。
   (5) **選択した問題**については，次の例に従って，**選択欄**の**問題番号**の (選) をマークしてください。マークがない場合は，採点の対象になりません。

〔問6と問10を選択した場合の例〕

選択欄					
問1	●	問6	●	問10	●
問2	●	問7	(選)	問11	(選)
問3	●	問8	(選)	問12	(選)
問4	●	問9	(選)	問13	(選)
問5	●				

   (6) 解答は，次の例題にならって，解答欄にマークしてください。

〔例題〕　次の［　　］に入れる正しい答えを，解答群の中から選べ。

春の情報処理技術者試験は，［ a ］月に実施される。

解答群

ア　2　　イ　3　　ウ　4　　エ　5

正しい答えは“ウ　4”ですから，次のようにマークしてください。

例題	a	(ア) (イ) ● (エ)

裏表紙の注意事項も，必ず読んでください。

共通に使用される擬似言語の記述形式

擬似言語を使用した問題では，各問題文中に注記がない限り，次の記述形式が適用されているものとする。

〔宣言，注釈及び処理〕

記述形式		説明
○		手続，変数などの名前，型などを宣言する。
/* 文 */		**文**に注釈を記述する。
処理	・**変数** ← **式**	**変数**に**式**の値を代入する。
	・**手続**(**引数**, …)	**手続**を呼び出し，**引数**を受け渡す。
	条件式 　**処理**	単岐選択処理を示す。 　**条件式**が真のときは**処理**を実行する。
	条件式 　**処理** 1 **処理** 2	双岐選択処理を示す。 　**条件式**が真のときは**処理** 1 を実行し，偽のときは**処理** 2 を実行する。
	条件式 　**処理**	前判定繰返し処理を示す。 　**条件式**が真の間，**処理**を繰り返し実行する。
	処理 **条件式**	後判定繰返し処理を示す。 　**処理**を実行し，**条件式**が真の間，**処理**を繰り返し実行する。
	変数：**初期値**，**条件式**，**増分** 　**処理**	繰返し処理を示す。 　開始時点で**変数**に**初期値**（定数又は式で与えられる）が格納され，**条件式**が真の間，**処理**を繰り返す。また，繰り返すごとに，**変数**に**増分**（定数又は式で与えられる）を加える。

〔演算子と優先順位〕

演算の種類	演算子	優先順位
単項演算	+，−，not	高 ↑
乗除演算	×，÷，%	
加減演算	+，−	
関係演算	＞，＜，≧，≦，＝，≠	
論理積	and	
論理和	or	↓ 低

注　整数同士の除算では，整数の商を結果として返す。%演算子は，剰余算を表す。

〔論理型の定数〕

true, false

次の問1から問5までの5問については，全問解答してください。

問1　機械語命令の実行に関する次の記述を読んで，設問に答えよ。

(1)　1語は16ビットで，命令語は1語長である。命令語の形式は図1のとおりである。

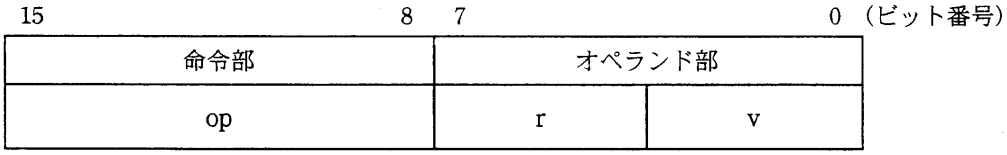


図1　命令語の形式

(2)　op，r及びvの意味を表1に示す。

レジスタは，1語長で16個（レジスタ番号 0～F）ある。op，r，v及びレジスタの内容は，16進数で表示する。

表1　op，r及びvの意味

	意味
op	命令コード（00 ≦ op ≦ FF）
r	レジスタ番号（0 ≦ r ≦ F）
v	レジスタ番号又は定数（0 ≦ v ≦ F）

(3)　命令の仕様（一部）を表2に示す。

表2　命令の仕様（一部）

名称	op（命令コード）	動作
LR (Load Register)	10	レジスタvの内容をレジスタrに設定する。
AR (Add Register)	20	レジスタrとレジスタvの内容の和を求め，結果をレジスタrに設定する。
OR (Or Register)	30	レジスタrとレジスタvの内容の論理和を求め，結果をレジスタrに設定する。
SL (Shift Left)	40	レジスタrの内容を定数vビットだけ左へシフトする。 シフトした結果，空いたビット位置には0を設定する。
SR (Shift Right)	50	レジスタrの内容を定数vビットだけ右へシフトする。 シフトした結果，空いたビット位置には0を設定する。

設問　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

レジスタ１～５の内容は，図２のとおりとする。

（レジスタ番号）	内容
1	1234
2	8361
3	5F2A
4	C38B
5	0010

図２　レジスタ１～５の内容

(1)　図２の状態で，次の命令語を実行するとレジスタ２の内容は［ a ］になる。

命令語：　3021

(2)　図２の状態で，次の４個の命令語を順に実行するとレジスタ３の内容は［ b ］に，レジスタ４の内容は［ c ］になる。

命令語：　1034
4038
5048
3043

(3)　図２の状態で，次の４個の命令語を順に実行するとレジスタ５の内容は10倍になる（レジスタ５：0010 → 00A0）。

命令語：　1065
4063
40［ d ］
2056

a に関する解答群

ア　9375	イ　9395	ウ　9575	エ　9595

b，c に関する解答群

ア　005F	イ　008B	ウ　00C3	エ　2A00
オ　2A5F	カ　8B00	キ　8BC3	ク　C38B

d に関する解答群

ア　51	イ　52	ウ　61	エ　62

問2　関係データベースに関する次の記述を読んで，設問1，2に答えよ。

ある情報システム開発会社は，顧客である A 社の組織情報を関係データベースにすることになり，まず，部門と所属についての設計を行った。

A 社の部門は，英字 2 文字からなる部門コードで一意に識別できる。A 社の社員は，4 けたの数字からなる社員番号で一意に識別でき，必ず一つの部門に所属しているものとする。

図 1 に A 社の部門と所属の情報を示す。括弧内は，部門コード又は社員番号である。

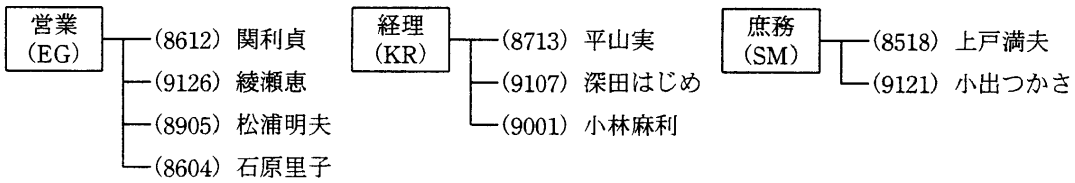


図 1　A 社の部門と所属の情報

設問1　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

情報システム開発会社の初級技術者 B 君は，案 1 と案 2 の二つの関係データベースの構造を考えた。しかし，上級技術者から，案 1 と案 2 の両方で，A 社の組織を表すには不都合な［ a ］という現象が起こり，案 1 では，更に［ b ］という不都合な現象も起こることが指摘された。

そこで B 君は，指摘に基づき，案 3 を考えた。

案 1 ～ 3 を，図 2 に示す。下線は，その項目が主キーであることを表す。

(案 1)

組織	氏名	社員番号	部門名	部門コード

(案 2)

組織	社員番号	氏名	部門コード	部門名

(案 3)

部門	部門コード	部門名

社員	社員番号	氏名	部門コード

図 2　案 1 ～ 3

解答群

ア　転属した社員の所属部門を，変更することができない

イ　新入社員を，登録することができない

ウ　退職した社員を，削除することができない

エ　同姓同名の社員を，登録することができない

オ　配属者未定の新設部門を，登録することができない

設問2　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

A社の制度が変更され，社員が本務として所属する部門に加え，その他の1部門までの兼務ができることになった。

このたび，新設部門として宣伝（SD）が設置され，本務が営業（EG）である綾瀬恵と本務が庶務（SM）である上戸満夫が，兼務として宣伝（SD）に配属されることになった。

B君は，これらに対応し，どの項目の内容も空にならない案4を考えた。

案4のデータベース構造において，今回の制度の変更，部門の新設及び配属の後，兼務の登録件数（行数）は，［ c ］となる。

案4を図3に示す。下線は，その項目が主キーであることを表す。

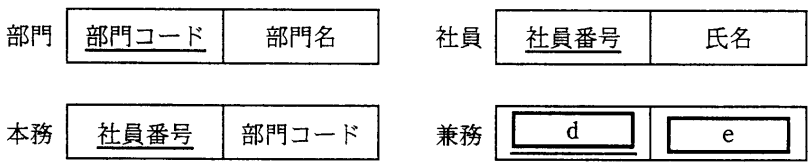


図3　案4

cに関する解答群

ア　1　　イ　2　　ウ　3　　エ　8　　オ　9

d，eに関する解答群

ア　氏名　　イ　社員番号　　ウ　部門コード

問3　システム開発の計画立案に関する次の記述を読んで，設問1，2に答えよ。

X社では，業務システムを新規に構築することになった。この業務システムには，新たなハードウェアの導入とソフトウェアの開発が必要である。ソフトウェアの開発では，業務プログラム内で共通に利用するプログラム（以下，共通部品という）を業務プログラム本体と分離して，別チームで開発することにした。今回の開発に関する作業の一覧と必要な日数を表に示す。

表　作業の一覧と必要な日数

作業ID	内容	日数
S1	システム方式設計	6
G1	業務プログラムの共通部品の定義	2
G2	業務プログラムの機能設計	4
G3	業務プログラムの詳細設計	6
G4	業務プログラムのコーディング・単体テスト	6
C1	共通部品の設計（インタフェース決定）	6
C2	共通部品のコーディング・単体テスト	4
H1	ハードウェアの調達	15
H2	ハードウェアの環境構築	6
S2	結合テスト	6
S3	実機テスト	4

この作業の日程計画を立てるために，図のアローダイアグラムを作成した。図中の破線は，ダミー作業である。

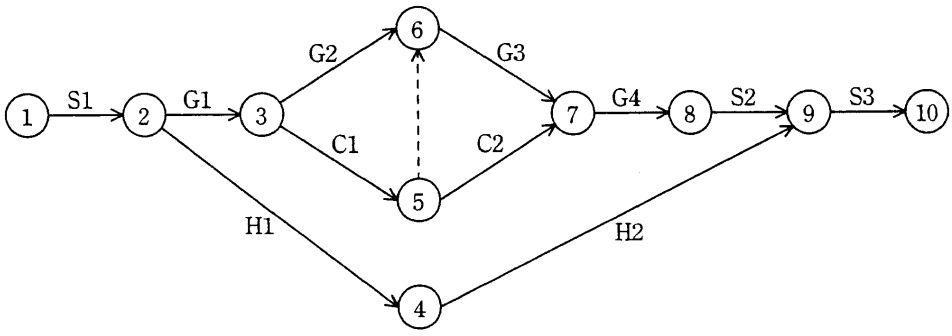


図　日程計画のアローダイアグラム

設問1　作業日程はアローダイアグラムに従うものとして，作業の依存関係に関する記述として正しい答えを，解答群の中から二つ選べ。

解答群

ア　業務プログラムの詳細設計（G3）は，共通部品のコーディング・単体テスト（C2）が終了しないと開始できない。

イ　業務プログラムのコーディング・単体テスト（G4）は，共通部品のコーディング・単体テスト（C2）が終了しないと開始できない。

ウ　業務プログラムの詳細設計（G3）は，業務プログラムの機能設計（G2）が終了すると開始できる。

エ　共通部品は実機がなくてもコーディング・単体テスト（C2）を開始できるが，業務プログラムは実機がなくてはコーディング・単体テスト（G4）を開始できない。

オ　ハードウェアの調達（H1）は，システム方式設計（S1）の完了後に行う。

設問2　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

この業務システムの開発におけるクリティカルパスは［ a ］，最短所要日数は［ b ］日である。また，全体の開発期間は，［ c ］を短縮することで短縮できる。

a に関する解答群

ア　①→②→③→⑤→⑥→⑦→⑧→⑨→⑩

イ　①→②→③→⑤→⑦→⑧→⑨→⑩

ウ　①→②→③→⑥→⑦→⑧→⑨→⑩

エ　①→②→④→⑨→⑩

b に関する解答群

ア　31	イ　32	ウ　33
エ　34	オ　35	カ　36

cに関する解答群

ア　共通部品の設計（C1）

イ　共通部品のコーディング・単体テスト（C2）

ウ　業務プログラムの機能設計（G2）

エ　ハードウェアの調達（H1）

問4　次のプログラムの説明及びプログラムを読んで，設問1～3に答えよ。

〔プログラムの説明〕

整数型の 1 次元配列の要素 A[0]，…，A[N]（N ＞ 0）を，挿入ソートで昇順に整列する副プログラム InsertSort である。

(1) 挿入ソートの手順は，次のとおりである。

① まず，A[0] と A[1] を整列し，次に A[0] から A[2] までを整列し，その次に A[0] から A[3] までというように，整列する区間の要素を一つずつ増やしていき，最終的に A[0] から A[N] までを整列する。

② 整列する区間が A[0] から A[M]（1 ≦ M ≦ N）までのとき，A[M] を既に整列された列 A[0]，…，A[M−1] 中の適切な位置に挿入する。その手順は次のとおりである。

(a) A[M] の値を，作業領域 Tmp に格納する。

(b) A[M−1] から A[0] に向かって Tmp と比較し，Tmp よりも大きな値を順次隣（要素番号の大きい方）に移動する。

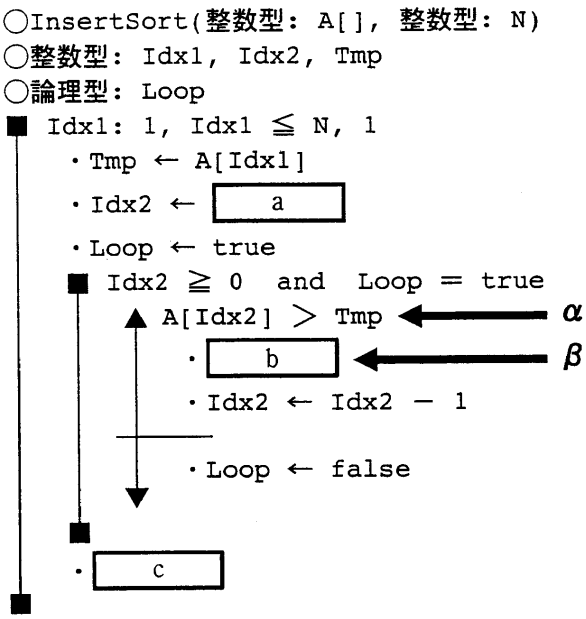
(c) (b) で最後に移動した値の入っていた配列要素に Tmp の内容を格納する。(b) で移動がなかった場合には A[M] に格納する。

(2) 副プログラム InsertSort の引数の仕様を表に示す。

表　InsertSort の引数の仕様

引数	データ型	入力／出力	意味
A[]	整数型	入出力	整列対象の 1 次元配列
N	整数型	入力	整列する区間の最後の要素番号

〔プログラム〕



設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a に関する解答群

ア　Idx1　　イ　Idx1 ＋ 1　　ウ　Idx1 − 1

b，c に関する解答群

ア　A[Idx2] ← A[Idx2＋1]　　イ　A[Idx2] ← A[Idx2−1]

ウ　A[Idx2] ← Tmp　　エ　A[Idx2＋1] ← A[Idx2]

オ　A[Idx2＋1] ← Tmp　　カ　A[Idx2−1] ← A[Idx2]

キ　A[Idx2−1] ← Tmp

設問2　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

1次元配列A[]の内容例を図に示す。

プログラム中のβの行が実行される回数は，図の例1では［　d　］回，例2では［　e　］回となる。

また，プログラム中のαの条件式をA[Idx2] ≧ Tmpとした場合，例3のように配列要素の中に同じ値が複数含まれるときには［　f　］。

A[]の要素番号	0	1	2	3	4	5	6
例1	0	1	4	3	2	5	6
例2	6	5	4	3	2	1	0
例3	3	3	1	5	4	5	2

図　1次元配列A[]の内容例

d，eに関する解答群

ア　2　　イ　3　　ウ　4

エ　19　　オ　20　　カ　21

fに関する解答群

ア　整列が正しく行われなくなる

イ　整列は正しく行われ，元の条件式の場合と比べて実行ステップ数は多くなる

ウ　整列は正しく行われ，元の条件式の場合と比べて実行ステップ数は変わらない

エ　整列は正しく行われ，元の条件式の場合と比べて実行ステップ数は少なくなる

設問3　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

n 個のランダムなデータを整列する場合，挿入ソートの計算量は $O(n^2)$，クイックソートの平均的な計算量は $O(n \log_2 n)$ なので，n の値が大きくなるとクイックソートの計算量の方が総じて小さくなる。

`InsertSort` のほかに，クイックソートで昇順に整列する副プログラム `QuickSort` を作成し，ランダムな 1,000 個のデータを何組か用意して，それぞれの組に対して `InsertSort` と `QuickSort` の実行時間を測定したところ，`QuickSort` の平均実行時間が `InsertSort` の $\frac{1}{10}$ であった。この結果を基にして，1,000,000 個のデータを整列したときの平均的な実行時間を計算すると，`QuickSort` が `InsertSort` の約 $\frac{1}{\boxed{\text{g}}}$ と推定できる。ここで，$\log_2 1000 \fallingdotseq 10$，$\log_2 1000000 \fallingdotseq 20$ とする。

解答群

ア　500	イ　1,000	ウ　5,000
エ　10,000	オ　50,000	カ　500,000

問5　プログラム設計に関する次の記述を読んで，設問1～3に答えよ。

ある会員情報が入ったマスタファイルの内容を，毎月末にトランザクションファイルの内容によって更新し，新マスタファイルに出力する更新プログラムを作成する。

〔ファイルの説明〕

(1) マスタファイル及びトランザクションファイルのレコード様式は同じで，次の項目からなる。

会員番号	名前	電子メールアドレス	サービス等級

(2) 会員番号は，5けたの数字であり，必須の項目である。最大値99999の会員番号をもつ会員は存在しない。

(3) マスタファイル及びトランザクションファイルは，会員番号をキーとして，昇順に整列されている。

(4) マスタファイルには，同一の会員番号をもつレコードが複数存在することはない。トランザクションファイルにも，同一の会員番号をもつレコードが複数存在することはない。

(5) トランザクションファイルは，新規会員追加に用いるレコード及び既に登録されている会員の会員情報変更に用いるレコードからなる。

(6) 会員情報変更に用いるレコードは，変更する項目に空白以外のデータが格納され，変更しない項目には，空白が格納されている。

〔処理の説明〕

マスタファイルのレコードをM，トランザクションファイルのレコードをTとして，次のキー突合せ処理を行う。

(1) Mと同一の会員番号をもつTがあるとき，Tの空白以外の項目でMの対応する項目を更新し，Tの空白の項目に対応したMの項目は更新せずに新マスタファイルに出力する。

(2) Mと同一の会員番号をもつTがないとき，Mをそのまま新マスタファイルに出力する。

(3) Tと同一の会員番号をもつMがないとき，Tを新マスタファイルに出力する。

更新プログラムの流れを図1に示す。

突合せキー K_M 及び K_T には，それぞれ，M の会員番号の値及び T の会員番号の値，又はそれぞれのファイルを読み終わったことを表す最大値 99999 を格納する。

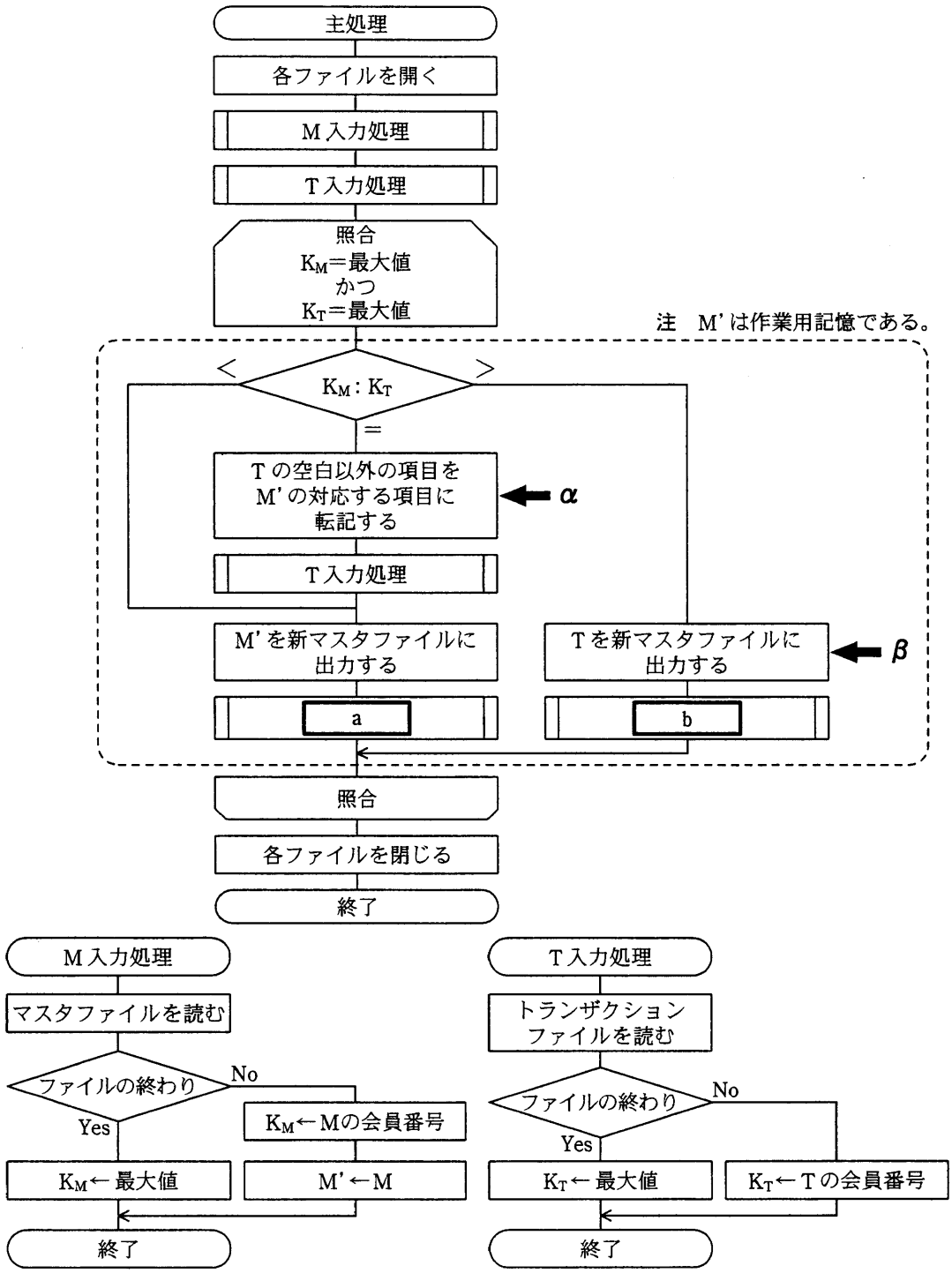


図 1　更新プログラムの流れ

設問1　図1において，キー突合せによるマスタファイルの更新処理は，破線で囲んだ部分で実現している。

図1中の［　　］に入れる正しい答えを，解答群の中から選べ。

解答群

ア　M'入力処理　　イ　M入力処理　　ウ　T入力処理

設問2　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

この更新プログラムに会員情報の削除処理を追加するため，次の(1)～(5)を行う。

(1) トランザクションファイルのレコード様式に，更新区分という項目を追加する。

(2) 更新区分が“U”の場合を新規登録及び変更とし，“D”の場合を削除とする。

(3) 図1の**α**と**β**の処理を，(1)のレコード様式の変更に対応するように変更する。

(4) 主処理を図2のとおりに変更する。

(5) T入力処理については，ファイルを読み終わったとき，更新区分に“U”を代入する処理を追加する。

ここで，図2のエラー処理1及び2は，エラーとなったレコードに関する情報を表示する。条件X2は，［ c ］である。エラー処理2は，［ d ］場合に実行される。

cに関する解答群

ア　更新区分＝“D”

イ　更新区分＝“U”

ウ　更新区分≠“D”

エ　更新区分≠“U”

オ　（更新区分≠“D”）AND（更新区分≠“U”）

カ　（更新区分≠“D”）OR（更新区分≠“U”）

dに関する解答群

ア　更新区分に誤りがある

イ　削除しようとする会員番号をもつレコードがマスタファイルに存在しない

ウ　新規登録しようとする会員番号をもつレコードがマスタファイルに存在する

エ　変更しようとする会員番号をもつレコードがマスタファイルに存在しない

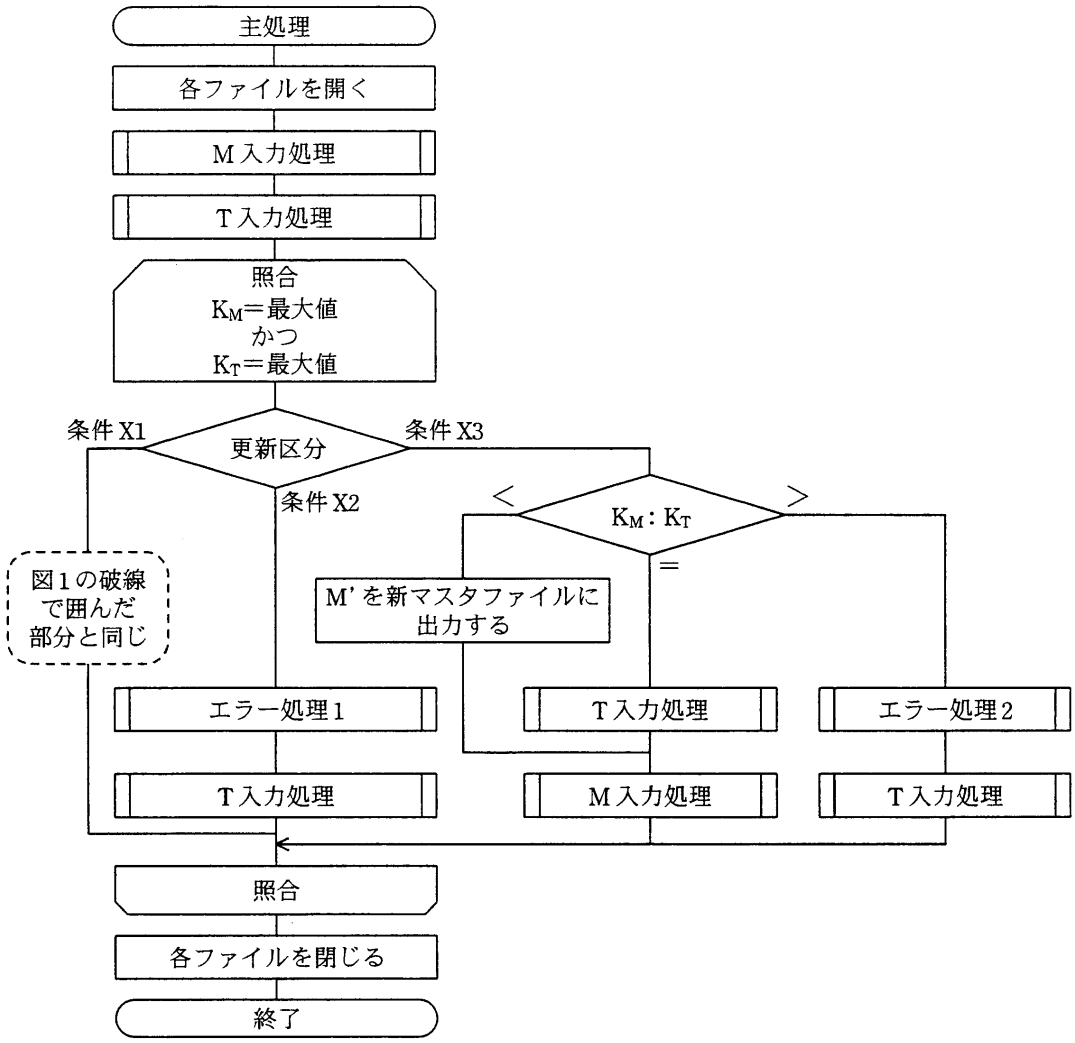


図2　変更後の更新プログラムの流れ

設問3　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

トランザクションファイル中に同一の会員番号をもつレコードが複数あってもよいようにする。複数ある場合，レコードの発生順に処理する。そのために，次の(1)，(2)の変更を行うことにした。

(1)　トランザクションファイルのレコード様式に発生日時という項目を追加する。

(2)　図2の更新プログラムとは別にトランザクションファイルの処理に，整列プログラムとレコード集約プログラムを追加して，新たにトランザクションファイルを生成する。

整列プログラムは，レコードを整列キーの昇順に並べ替える。ここで，第1整列キーは［ e ］，第2整列キーは［ f ］である。

レコード集約プログラムの主な機能は，次の二つである。

① 会員番号が同じ入力レコードで，更新区分がすべて“U”のときには，空白以外の変更項目の内容を作業領域に上書きし，最後の状態を出力する。

② 更新区分が“D”のレコードが見つかったときには，そのレコードを出力する。それ以降にも同一の会員番号のレコードが存在したときには，それらは処理せずエラーとする。

これらのプログラムの実行順序は，次のとおりである。

実行順序：［ g ］

e，fに関する解答群

ア 会員番号　　イ 更新区分　　ウ サービス等級　　エ 発生日時

gに関する解答群

ア 更新プログラム → 整列プログラム → レコード集約プログラム

イ 更新プログラム → レコード集約プログラム → 整列プログラム

ウ 整列プログラム → 更新プログラム → レコード集約プログラム

エ 整列プログラム → レコード集約プログラム → 更新プログラム

オ レコード集約プログラム → 更新プログラム → 整列プログラム

カ レコード集約プログラム → 整列プログラム → 更新プログラム

次の問6から問9までの 4 問については，この中から 1 問を選択し，答案用紙の選択欄の (選) をマークして解答してください。

なお，2 問以上選択した場合には，はじめの 1 問について採点します。

問6　次の C プログラムの説明及びプログラムを読んで，設問 1，2 に答えよ。

〔プログラムの説明〕

(1) 双六(すごろく)のプログラムである。双六とは，図のような一本道のコース上にある升(ます)を，“ふりだし”の位置からスタートして，プレーヤが順番にさいころ 2 個を振って，出た目の和だけ駒(こま)を進め，早く“上がり”（ゴール）の位置にたどり着いたプレーヤを勝ちとする遊びである。

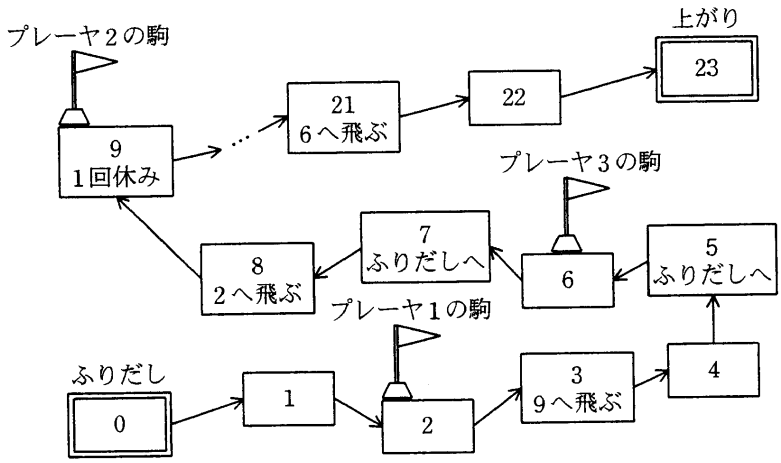


図　双六の例

(2) 双六の進め方は，次のとおりである。

① プレーヤは“ふりだし”の升に各自の駒を置き，さいころを振る順番を決める。

② 順番にさいころを振って，出た目の和だけ“上がり”方向に駒を進める。

③ “上がり”の升に到達すると，そのプレーヤの勝ちとなり，そこでプログラムが終了する。目の和だけ駒を進めた結果が“上がり”の升を超えた場合も，“上がり”とする。

④ 升には順番に番号が割り当てられている。番号 0 の升が“ふりだし”であり，番号 23 の升が“上がり”である。

⑤ 升には番号のほかに，到達時の動作指示が割り当てられているものがある。
動作指示には，次の2種類があり，その指示に従う。

(a) 移動の指示： 指示された番号の升に駒を移動する。移動先の升に動作指示がある場合は，その指示に従う。

(b) 1回休みの指示： 次に順番が回ってきたとき，“さいころを振る番”を1回休む。

(3) プレーヤは，4人とする。

(4) 動作指示は，配列 actype 及び celinf に設定されている。

(5) 次の関数が，あらかじめ定義されているものとする。

① 関数 dice

機能： 引数で指定されたプレーヤに対してさいころを振る番が来たことを示すメッセージを表示する。プレーヤが改行キーを入力すると，乱数によって2個のさいころの目の和を決めて，その値を表示し，返却値とする。

呼出し形式： `int dice(int playerid);`

引数： `playerid` プレーヤ番号（0 ～ 3）

返却値： さいころの目の和（2 ～ 12 の整数値）

② 関数 prtpiece

機能： 引数で指定されたプレーヤの駒が現在置かれている升の番号と，升に割り当てられている動作指示の内容を表示する。

呼出し形式： `void prtpiece(int playerid);`

引数： `playerid` プレーヤ番号（0 ～ 3）

返却値： なし

〔プログラム〕

```
#include <stdio.h>
#define CELLS   24
#define PLAYERS 4
#define PLAY    1
#define WAIT    0

int dice(int);
void prtpiece(int);

static int actype[CELLS] = { 0, 0, 0, 1, 0, 1, 0, 1, 1, 2, 0, 0,
                             0, 0, 0, 2, 0, 0, 0, 1, 0, 1, 0, 0 };
static int celinf[CELLS] = { 0, 0, 0, 9, 0, 0, 0, 0, 2, 0, 0, 0,
                             0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0 };
static int curpos[PLAYERS], playmode[PLAYERS];

void main()
{
   int gamestatus = 1, playerid;

   printf("双六を開始します。\n");
   for(playerid = 0; playerid < PLAYERS; playerid++){
      curpos[playerid] = 0;
      playmode[playerid] = PLAY;
   }
   playerid = 0;
   while(gamestatus == 1){
      if(playmode[playerid] == PLAY){
         curpos[playerid] += dice(playerid);
         prtpiece(playerid);
         while((curpos[playerid] < CELLS - 1) && ( [ a ] )
               && (playmode[playerid] == PLAY)){
            switch(actype[curpos[playerid]]){
               case  1:   /* 指示先の升へ飛ぶ。 */
                  curpos[playerid] = celinf[curpos[playerid]];
                  prtpiece(playerid);
                  break;
               case  2:   /* 1回休み */
                  playmode[playerid] = WAIT;
                  break;
            }
         }
         if( [ b ] ) gamestatus = 0;
      }
      else playmode[playerid] = PLAY;
      if(gamestatus == 1) [ c ];
   }
   printf("プレーヤ ID:%d の勝ちです。\n", playerid);
}
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a，bに関する解答群

ア　actype[curpos[playerid]] != 0

イ　actype[curpos[playerid]] <= 3

ウ　curpos[playerid] == 0

エ　curpos[playerid] >= CELLS - 1

オ　gamestatus == 0

カ　gamestatus == 1

キ　playmode[playerid] == PLAY

ク　playmode[playerid] == WAIT

cに関する解答群

ア　playerid = playerid % PLAYERS + 1

イ　playerid = playerid % (PLAYERS + 1)

ウ　playerid = (playerid + 1) % PLAYERS

エ　playerid += 1

オ　playerid += playerid / (PLAYERS - 1)

設問2　到達時の動作指示として，“続けてさいころを振り，駒を進める”を追加することにした。そのためにswitch文でのactype[curpos[playerid]]の値が3のときに，次の処理群を挿入する。［　　］に入れる正しい答えを，解答群の中から選べ。

```
case  3:  /* 続けてさいころを振り，駒を進める。 */
    [          d          ];
    prtpiece(playerid);
    break;
```

解答群

ア　curpos[playerid] = celinf[curpos[playerid]]

イ　curpos[playerid] = celinf[curpos[playerid]] + dice(playerid)

ウ　curpos[playerid] += celinf[curpos[playerid] + dice(playerid)]

エ　curpos[playerid] += dice(playerid)

オ　playmode[playerid] = PLAY

問7　次のCOBOLプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

〔プログラムの説明〕

選挙速報を画面に表示するプログラムである。あるテレビ局では，地方自治体の首長選挙において，地区ごとの開票情報を収集し，候補者ごとの得票数を集計して表示するプログラムを開発した。

(1) 地区ごとの開票情報を記録した開票ファイルのレコード様式は，次のとおりである。

地区コード 6けた	候補者名 20けた	得票数 9けた

(2) 選挙速報の表示様式は，次のとおりである。

候補者名	得票数
XXX … XXX	ZZZ,ZZZ,ZZ9
⋮	⋮

① 選挙速報は，得票数の降順に表示する。

② 見出しは，あらかじめ表示されている。

(3) 候補者数は最大50人，得票数は最大9けたとし，データに誤りはないものとする。

〔プログラム〕

```
DATA DIVISION.
FILE SECTION.
FD  KAIHYO-F.
01  KAIHYO-R.
    05  K-CHIKU-CD              PIC X(6).
    05  K-KOHOSHA-MEI           PIC X(20).
    05  K-TOKUHYO-SU            PIC 9(9).
WORKING-STORAGE SECTION.
01  SHUKEI-TABLE.
    05  S-MAX                   PIC 9(3).
    05  S-TBL    OCCURS 0 TO 50 DEPENDING ON S-MAX
                 INDEXED BY S-IDX.
        10  S-KOHOSHA-MEI       PIC X(20).
        10  S-TOKUHYO-SU        PIC 9(9).
```

```
 01  W-TBL                      PIC X(29).
 01  P-TOKUHYO-SU              PIC ZZZ,ZZZ,ZZ9.
 01  W-EOF                     PIC 9.
 01  W-I                       PIC 9(5).
 01  W-J                       PIC 9(5).
 01  W-K                       PIC 9(5).
 PROCEDURE DIVISION.
 MAIN-CTL.
     OPEN INPUT KAIHYO-F.
     MOVE 0 TO W-EOF.
     MOVE 0 TO S-MAX.
     PERFORM UNTIL W-EOF = 1
        READ KAIHYO-F
             AT END      MOVE 1 TO W-EOF
             NOT AT END  PERFORM SHUKEI-PROC
        END-READ
     END-PERFORM.
     PERFORM SHUKEI-DISP.
     CLOSE KAIHYO-F.
     STOP RUN.
*
 SHUKEI-PROC.
     SET S-IDX TO 1.
     SEARCH S-TBL VARYING S-IDX
        AT END
             ADD 1 TO S-MAX
             MOVE K-KOHOSHA-MEI TO S-KOHOSHA-MEI(S-MAX)
             [          a          ]
        WHEN S-KOHOSHA-MEI(S-IDX) = K-KOHOSHA-MEI
             [          b          ]
     END-SEARCH.
*
 SHUKEI-DISP.
     PERFORM VARYING W-I FROM 1 BY 1 UNTIL W-I > S-MAX
        COMPUTE W-K = W-I + 1
        PERFORM VARYING W-J FROM W-K BY 1 UNTIL W-J > S-MAX
           IF S-TOKUHYO-SU(W-I) < S-TOKUHYO-SU(W-J)
              MOVE S-TBL(W-I) TO W-TBL
              [          c          ]
              MOVE W-TBL TO S-TBL(W-J)
           END-IF
        END-PERFORM
        MOVE S-TOKUHYO-SU(W-I) TO P-TOKUHYO-SU
        DISPLAY S-KOHOSHA-MEI(W-I) "    " P-TOKUHYO-SU
     END-PERFORM.
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

解答群

ア　ADD K-TOKUHYO-SU TO S-TOKUHYO-SU(S-IDX)

イ　ADD K-TOKUHYO-SU TO S-TOKUHYO-SU(S-MAX)

ウ　MOVE K-TOKUHYO-SU TO S-TOKUHYO-SU(S-MAX)

エ　MOVE S-TBL(W-I) TO S-TBL(W-J)

オ　MOVE S-TBL(W-J) TO S-TBL(W-I)

設問2　このプログラムの動作を検証するための命令網羅テストにおいて，開票ファイルとして適切なデータの並びを，解答群の中から選べ。

解答群

ア

地区コード	候補者名	得票数
AAAAAA	WWWWWW	1000
AAAAAA	XXXXXX	2000
CCCCCC	XXXXXX	3000
CCCCCC	ZZZZZZ	4000
EEEEEE	VVVVVV	3000
EEEEEE	WWWWWW	5000

イ

地区コード	候補者名	得票数
AAAAAA	UUUUUU	1000
BBBBBB	UUUUUU	2000
CCCCCC	UUUUUU	3000
DDDDDD	UUUUUU	4000
EEEEEE	UUUUUU	5000
FFFFFF	UUUUUU	3000

ウ

地区コード	候補者名	得票数
BBBBBB	UUUUUU	1000
BBBBBB	VVVVVV	2000
BBBBBB	WWWWWW	3000
BBBBBB	XXXXXX	4000
BBBBBB	YYYYYY	2000
BBBBBB	ZZZZZZ	9000

エ

地区コード	候補者名	得票数
BBBBBB	UUUUUU	1000
BBBBBB	XXXXXX	2000
CCCCCC	WWWWWW	3000
CCCCCC	VVVVVV	4000
DDDDDD	YYYYYY	7000
DDDDDD	ZZZZZZ	3000

オ

地区コード	候補者名	得票数
BBBBBB	WWWWWW	1000
BBBBBB	XXXXXX	2000
DDDDDD	XXXXXX	3000
DDDDDD	YYYYYY	4000
FFFFFF	WWWWWW	3000
FFFFFF	YYYYYY	5000

問8　次のJavaプログラムの説明及びプログラムを読んで，設問に答えよ。

（Javaプログラムで使用するAPIの説明は，この冊子の末尾を参照してください。）

〔プログラムの説明〕

英語の月名（January，…，December）を2種類の順序に並べ替えるプログラムである。月名は，文書の索引などを作成する場合は辞書順に並べ替える必要があるが，日付の一部として並べ替えるときは月の順番に並んだ方が都合がよい。並べ替えは，次のクラスメソッドを呼び出して行う。

```
java.util.Arrays.sort(Object[], java.util.Comparator)
```

このメソッドは，引数で与えられた Comparator のインスタンスのメソッド compare を呼び出して配列中の要素を比較し，並べ替える。ここでは，インタフェース Comparator を実装する次の二つのクラスを定義する。

(1) クラス NameComparator は，引数で与えられた文字列を小文字の文字列に変換し，クラス String のメソッド compareTo を呼び出して比較した結果を返すメソッドを実装している。

(2) クラス ValueComparator は，引数で与えられた文字列（月名）を月の値（例えば，April は4）に変換し，月の値で比較した結果を返すメソッドを実装している。引数の文字列は，大文字と小文字を区別しない。月名は，Jan，Feb など，3文字の省略形を指定してもよい。

ただし，二つのクラスともメソッド compare に与えられる月名は正しいものとする。

クラス MonthNameSorter は，上記の二つの Comparator を実装したクラスを用いて，月名を並べ替える。メソッド main を実行した結果を，次に示す。

```
[December, July, DEC, June, April, May]
[April, DEC, December, July, June, May]
[April, May, June, July, December, DEC]
```

〔プログラム1〕

```
import java.util.Comparator;
import java.util.Locale;

public class NameComparator implements Comparator<String> {
   public int compare(String s1, String s2) {
      String ls1 = s1.toLowerCase(Locale.ENGLISH);
      String ls2 = s2.toLowerCase(Locale.ENGLISH);
      return ls1.compareTo(ls2);
   }
}
```

〔プログラム2〕

```
import java.util.Comparator;
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;

public class ValueComparator implements [      a      ] {
   private static final String[] monthNames = {
      "january", "february", "march", "april",
      "may", "june", "july", "august",
      "september", "october", "november", "december"
   };
   private static final Map<String, Integer> map
                                 = new HashMap<String, Integer>();
   static {
      // 月名とその値（例：may → 5）が対応するようにmapを初期化する。
      for (int i = 0; [      b      ]; i++) {
         // 月名と月の値を対応付ける。
         map.put(monthNames[i], i + 1);
         // 月名の最初の3文字と月の値を対応付ける。
         map.put(monthNames[i].substring(0, 3), i + 1);
      }
   }

   public int compare(String s1, String s2) {
      String ls1 = s1.toLowerCase(Locale.ENGLISH);
      String ls2 = s2.toLowerCase(Locale.ENGLISH);
      return [      c      ];
   }
}
```

Java

〔プログラム3〕

```
import java.util.Arrays;

public class MonthNameSorter {
   public static void main(String[] args) {
      final String[] names1 = {
         "December", "July", "DEC", "June", "April", "May"
      };
      System.out.println(Arrays.toString(names1));

      String[] names2 = names1.clone();
      Arrays.sort(names2, new NameComparator());
      System.out.println(Arrays.toString(names2));

      String[] names3 = names1.clone();
      Arrays.sort(names3, [ d ]);
      System.out.println(Arrays.toString(names3));
   }
}
```

設問　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a に関する解答群

ア　Comparator<int>　　イ　Comparator<Integer>

ウ　Comparator<Object>　　エ　Comparator<String>

b に関する解答群

ア　i < monthNames.length　　イ　i <= monthNames.length

ウ　i > monthNames.length　　エ　i >= monthNames.length

c に関する解答群

ア　map.get(ls1) + map.get(ls2)　　イ　map.get(ls1) - map.get(ls2)

ウ　map.get(ls1) / map.get(ls2)　　エ　map.get(ls2) - map.get(ls1)

d に関する解答群

ア　new Comparator()　　イ　new Comparator<String>()

ウ　new ValueComparator()　　エ　new ValueComparator<String>()

問9　次のアセンブラプログラムの説明及びプログラムを読んで，設問 1 ～ 3 に答えよ。

〔プログラムの説明〕

副プログラム SYMTST は，16 ビットからなるビット列が左右対称かどうかを検査するプログラムである。図に左右対称なビット列の例を示す。

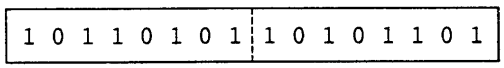


図　左右対称なビット列の例

(1) ビット列は GR1 に設定されて，主プログラムから渡される。

(2) ビット列が左右対称であれば GR0 に 1 を，そうでなければ 0 を設定して主プログラムに返す。

(3) 副プログラムから戻るとき，汎用レジスタ GR1 ～ GR7 の内容は元に戻す。

〔プログラム〕

(行番号)

```
 1  SYMTST  START
 2          RPUSH
 3          LD     GR3,=8
 4          LD     GR2,GR1
 5  LOOP    SLL    GR1,1
 6          JOV    OFLOW
 7          SRL    GR2,1          ; 右端のビットは 1 か？
 8          JOV    NG             ; 1 であれば左右対称でない。
 9          JUMP   OK
10  OFLOW   SRL    GR2,1
11          [      a      ]
12  NG      LD     GR0,=0         ; 左右対称でない。
13          JUMP   FIN
14  OK      SUBA   GR3,=1
15          [      b      ]
16          LD     GR0,=1         ; 左右対称である。
17  FIN     RPOP
18          RET
19          END
```

アセンブラ

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

解答群

ア	JMI	LOOP	イ	JOV	NG	ウ	JOV	OFLOW
エ	JOV	OK	オ	JPL	LOOP	カ	JZE	LOOP

設問2　ビット列が図のとおりであった場合，行番号 17 のマクロ命令 RPOP の実行直前における GR2 の内容として正しい答えを，解答群の中から選べ。

解答群

ア

0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

イ

0 0 0 0 0 0 0 0	1 0 1 1 0 1 0 1

ウ

0 0 0 0 1 0 1 1	0 1 0 1 1 0 1 0

エ

1 0 1 0 1 1 0 1	0 0 0 0 0 0 0 0

設問3　ビット列が次のとおりであった場合（▲で示されるビットが左右対称ではない場合），行番号 10 の SRL 命令は何回実行されるか。正しい答えを，解答群の中から選べ。

1 0 1 1 0 1 0 1	1 0 1 0 0 1 0 1
▲	▲

解答群

ア　1　　イ　2　　ウ　3　　エ　4

次の**問10**から**問13**までの4問については，この中から1問を選択し，答案用紙の選択欄の(選)をマークして解答してください。

なお，2問以上選択した場合には，はじめの1問について採点します。

問10 次のCプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

〔プログラムの説明〕

リーグ戦の勝敗表を出力するプログラムである。

(1) 勝敗表の出力例を図1に示す。

チーム名	勝ち	負け	勝率
Bishops	27	23	0.540
Kings	33	17	0.660
Knights	27	23	0.540
Pawns	12	38	0.240
Queens	32	18	0.640
Rooks	19	31	0.380

図1 勝敗表の出力例

(2) チームの勝敗情報は構造体RECORDで表現する。引き分けはないものとする。

```
typedef struct {                    /* 勝敗情報 */
    char   name[MAX_LENGTH+1]; /* チーム名 */
    int    wins;                    /* 勝ち数 */
    int    losses;                  /* 負け数 */
    double average;                 /* 勝率 */
} RECORD;
```

全チームの勝敗情報は構造体RECORDの配列teamに格納されている。チーム名，勝ち数，負け数はあらかじめ格納されているが，勝率は格納されていない。

(3) プログラム中の関数calcAverageとprintの仕様は次のとおりである。

`void calcAverage()`

機能：全チームの勝率を計算し，それぞれのメンバaverageに格納する。

試合数が0の場合，勝率は0.0とする。

`void print()`

機能：勝敗表を出力する。

〔プログラム1〕

```
#include <stdio.h>
#define TEAMNUM 6                   /* チーム数 */
#define MAX_LENGTH 10               /* チーム名の長さの上限 */

typedef struct {                    /* 勝敗情報 */
    char   name[MAX_LENGTH+1]; /* チーム名 */
    int    wins;                /* 勝ち数 */
    int    losses;              /* 負け数 */
    double average;             /* 勝率 */
} RECORD;

void calcAverage();
void print();

RECORD team[TEAMNUM];

void calcAverage(){
    int i, total;
    for(i = 0; i < TEAMNUM; i++){
        total = team[i].wins + team[i].losses;
        if( [ a ] ){
            team[i].average = 0.0;
        } else {
            [ b ];
        }
    }
}

void print(){
    int i;
    calcAverage();
    printf("チーム名    勝ち  負け      勝率\n");
    for(i = 0; i < TEAMNUM; i++){
        printf("%-10s %4d  %4d    %5.3f\n",
          team[i].name, team[i].wins, team[i].losses, team[i].average
    }
}
```

設問1　プログラム1中の［　　］に入れる正しい答えを，解答群の中から選べ。

aに関する解答群

ア　team[i].average < 0.0　　イ　team[i].average > 0.0

ウ　total != 0　　エ　total == 0

b に関する解答群

ア (double)team[i].average = team[i].wins / total

イ team[i].average = (double)(team[i].wins / total)

ウ team[i].average = (double)team[i].wins / total

エ team[i].average = team[i].wins / total

設問 2 順位の項目を加え，勝率の高いチームから順に出力する関数 printRank を作成した。勝率順の勝敗表の出力例を図 2 に示す。勝率が同率の場合は同順位とする。

C

順位	チーム名	勝ち	負け	勝率
1	Kings	33	17	0.660
2	Queens	32	18	0.640
3	Bishops	27	23	0.540
3	Knights	27	23	0.540
5	Rooks	19	31	0.380
6	Pawns	12	38	0.240

図 2 勝率順の勝敗表の出力例

処理手順は次のとおりである。

(1) TEAMNUM 個の要素をもつポインタ配列 pTeam を定義する。

(2) 配列 team の各要素のアドレスを，先頭要素から順番に配列 pTeam の各要素に格納する。これによって要素番号 i に対応するチームの勝率 team[i].average は，pTeam[i]->average によっても参照できる。

(3) 配列 team 自体を整列する代わりに配列 pTeam の要素を整列して，勝率順の勝敗表を出力する。

次の関数があらかじめ定義されているものとする。

```
void sort(RECORD *pTeam[TEAMNUM])
```

機能：TEAMNUM 個の要素からなり，各要素が RECORD 型の構造体へのポインタである配列 pTeam の要素を，勝率の降順になるように整列する。

プログラム 2 中の [　　] に入れる正しい答えを，解答群の中から選べ。

〔プログラム2〕

```
void printRank();
void sort(RECORD *[TEAMNUM]);

void printRank(){
    int i;
    int rank = 1;
    RECORD *pTeam[TEAMNUM];

    calcAverage();
    for(i = 0; i < TEAMNUM; i++){
        [ c ];
    }
    sort(pTeam);
    printf("順位   チーム名    勝ち  負け     勝率\n");
    for(i = 0; i < TEAMNUM; i++){
        if(i > 0){
            if(pTeam[i]->average != pTeam[i-1]->average){
                [ d ];
            }
        }
        printf("%1d      %-10s %4d  %4d    %5.3f\n", rank,
                [ e ]->name, [ e ]->wins,
                [ e ]->losses, [ e ]->average);
    }
}
```

c に関する解答群

ア　pTeam[i] = &team[i]　　イ　pTeam[i] = team

ウ　team[i] = **pTeam　　エ　team[i] = *pTeam[i]

d に関する解答群

ア　rank++　　イ　rank = i

ウ　rank = i + 1　　エ　rank = TEAMNUM - (i + 1)

オ　rank = TEAMNUM - i

e に関する解答群

ア　(&team[i])　　イ　(&team[rank])

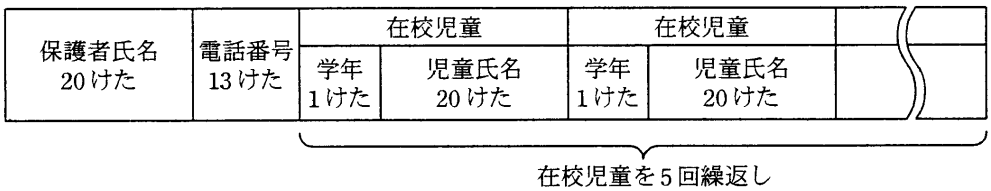
ウ　pTeam[i]　　エ　pTeam[rank]

問 11　次の COBOL プログラムの説明及びプログラムを読んで，設問 1，2 に答えよ。

〔プログラムの説明〕

ある小学校における保護者名簿の作成プログラムである。昨年度の保護者名簿及び新入学児童（新 1 年生）の保護者名簿を入力し，本年度の保護者名簿を作成する。

(1) 昨年度の保護者名簿及び本年度の保護者名簿は，次のレコード様式の順ファイルである。



① レコードは，先頭から 33 けたのデータ（保護者氏名及び電話番号）の昇順に整列されている。

② 在校児童欄には，保護者が保護する在校児童の学年と氏名が格納されている。

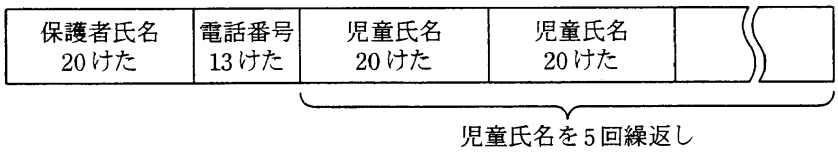
③ 在校児童が複数いる場合，学年で昇順に整列されている。

④ 在校児童が 4 人以下の場合，左詰めで格納され，残りの領域には空白が詰められている。

例：3 年生と 5 年生に在校児童をもつ保護者の場合，次のように格納される。

中村健	03-1234-5678	3	中村二郎	5	中村一郎	空白

(2) 新入学児童の保護者名簿は，次のレコード様式の順ファイルである。



① レコードは，先頭から 33 けたのデータ（保護者氏名及び電話番号）の昇順に整列されている。

② 児童氏名には，保護者が保護する新入学児童の氏名が格納されている。

③ 新入学児童が 4 人以下の場合，左詰めで格納され，残りの領域には空白が詰められている。

(3) 1 人の保護者が保護する児童は 5 人を超えないものとする。

(4) 在校児童の学年は無条件に1学年繰り上げ，6学年を修了した児童は卒業する。

(5) 転入や転出は考慮しない。

(6) 保護している在校児童がすべて卒業し，かつ保護する新入学児童がいない場合，その保護者のレコードを削除する。

(7) 新入学児童の保護者が未登録だった場合，その保護者のレコードを新規に登録する。

〔プログラム〕

(行番号)

```
 1  DATA DIVISION.
 2  FILE SECTION.
 3  FD  OLD-FILE.
 4  01  OLD-REC            PIC X(138).
 5  FD  NEW-FILE.
 6  01  NEW-REC            PIC X(138).
 7  FD  ENT-FILE.
 8  01  ENT-REC            PIC X(133).
 9  WORKING-STORAGE SECTION.
10  01 W-OLD-REC.
11     02 OLD-ID           VALUE SPACE.
12        88 OLD-EOF       VALUE HIGH-VALUE.
13           03 OLD-PARENT      PIC X(20).
14           03 OLD-TEL         PIC X(13).
15     02 OLD-PUPIL              OCCURS 5 TIMES.
16        03 OLD-NUM             PIC 9(1).
17        03 OLD-NAME            PIC X(20).
18  01 W-NEW-REC.
19     02 NEW-ID.
20        03 NEW-PARENT          PIC X(20).
21        03 NEW-TEL             PIC X(13).
22     02 NEW-PUPIL              OCCURS 5 TIMES.
23        03 NEW-NUM             PIC 9(1).
24        03 NEW-NAME            PIC X(20).
25  01 W-ENT-REC.
26     02 ENT-ID           VALUE SPACE.
27        88 ENT-EOF       VALUE HIGH-VALUE.
28           03 ENT-PARENT      PIC X(20).
29           03 ENT-TEL         PIC X(13).
30     02 ENT-NAME               OCCURS 5 TIMES PIC X(20).
31  01 OLD-CNT                  PIC 9(1).
32  01 NEW-CNT                  PIC 9(1).
33  01 ENT-CNT                  PIC 9(1).
34  01 READ-FLAG                PIC X(1) VALUE "B".
35     88 READ-BOTH             VALUE "B".
36     88 READ-OLD-FILE         VALUE "O".
37     88 READ-ENT-FILE         VALUE "E".
```

COBOL

```
38  PROCEDURE DIVISION.
39  MAIN-PROC.
40      OPEN INPUT OLD-FILE ENT-FILE OUTPUT NEW-FILE.
41      PERFORM UNTIL OLD-EOF AND ENT-EOF
42         IF READ-OLD-FILE OR READ-BOTH THEN
43            READ OLD-FILE INTO W-OLD-REC
44                          AT END SET OLD-EOF TO TRUE
45            END-READ
46         END-IF
47         IF READ-ENT-FILE OR READ-BOTH THEN
48            READ ENT-FILE INTO W-ENT-REC
49                          AT END SET ENT-EOF TO TRUE
50            END-READ
51         END-IF
52         IF [          a          ] THEN
53            PERFORM CREATE-PROC
54         END-IF
55      END-PERFORM.
56      CLOSE OLD-FILE ENT-FILE NEW-FILE.
57      STOP RUN.
58  CREATE-PROC.
59      MOVE SPACE TO W-NEW-REC.
60      EVALUATE TRUE
61      WHEN OLD-ID < ENT-ID
62         PERFORM NUM-UP-PROC
63         [          b          ]
64         SET READ-OLD-FILE TO TRUE
65      WHEN OLD-ID = ENT-ID
66         PERFORM NUM-UP-PROC
67         MOVE OLD-ID TO NEW-ID
68         PERFORM ENT-ADD-PROC
69         MOVE 1 TO OLD-CNT
70         PERFORM VARYING NEW-CNT FROM ENT-CNT BY 1
71                 UNTIL NEW-CNT > 5
72            [          c          ]
73            ADD 1 TO OLD-CNT
74         END-PERFORM
75         SET READ-BOTH TO TRUE
76      WHEN OLD-ID > ENT-ID
77         MOVE ENT-ID TO NEW-ID
78         PERFORM ENT-ADD-PROC
79         SET READ-ENT-FILE TO TRUE
80      END-EVALUATE.
81      IF NEW-PUPIL(1) NOT = SPACE THEN
82         WRITE NEW-REC FROM W-NEW-REC
83      END-IF.
```

```
84  NUM-UP-PROC.
85      PERFORM VARYING OLD-CNT FROM 1 BY 1
86              UNTIL OLD-CNT > 5 OR OLD-PUPIL(OLD-CNT) = SPACE
87          IF OLD-NUM(OLD-CNT) < 6 THEN
88              [          d          ]
89          ELSE
90              MOVE SPACE TO OLD-PUPIL(OLD-CNT)
91          END-IF
92      END-PERFORM.
93  ENT-ADD-PROC.
94      PERFORM VARYING ENT-CNT FROM 1 BY 1
95              UNTIL ENT-CNT > 5 OR ENT-NAME(ENT-CNT) = SPACE
96          MOVE 1 TO NEW-NUM(ENT-CNT)
97          MOVE ENT-NAME(ENT-CNT) TO NEW-NAME(ENT-CNT)
98      END-PERFORM.
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a に関する解答群

ア　NOT (OLD-EOF AND ENT-EOF)　　イ　NOT (OLD-EOF OR ENT-EOF)

ウ　OLD-EOF AND ENT-EOF　　エ　OLD-EOF OR ENT-EOF

b ～ d に関する解答群

ア　ADD 1 TO OLD-NUM(OLD-CNT)

イ　MOVE OLD-ID TO NEW-ID

ウ　MOVE OLD-PUPIL(ENT-CNT) TO NEW-PUPIL(NEW-CNT)

エ　MOVE OLD-PUPIL(OLD-CNT) TO NEW-PUPIL(NEW-CNT)

オ　MOVE W-OLD-REC TO W-NEW-REC

カ　PERFORM ENT-ADD-PROC

設問2　保護者名簿から削除される保護者，つまり，昨年度の保護者名簿には登録されていたが，本年度の保護者名簿には登録されない保護者の氏名と電話番号を画面に表示するようにプログラムを改造したい。正しい答えを，解答群の中から選べ。

解答群

ア

処置	プログラムの変更内容
行番号53と行番号54の間に追加	ELSE 　DISPLAY "REMOVED DATA = " OLD-ID

イ

処置	プログラムの変更内容
行番号82と行番号83の間に追加	ELSE 　DISPLAY "REMOVED DATA = " OLD-ID

ウ

処置	プログラムの変更内容
行番号90と行番号91の間に追加	DISPLAY "REMOVED DATA = " OLD-ID

エ

処置	プログラムの変更内容
行番号92と行番号93の間に追加	DISPLAY "REMOVED DATA = " OLD-ID.

問12　次のJavaプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

〔プログラムの説明〕

三目並べを行うプログラムである。

盤は3×3の升(ます)目をもち，2人のプレーヤは交互に自分の記号（o又はx）を書く。たて，横，斜めでの三つの升の並びを列と呼び，先に，どれかの列で三つの升すべてに，自分の記号を書いた方を勝ちとする。また，盤のすべての升に記号が書かれていて，どちらの勝ちでもないときは，引き分けとする。先手はo，後手はxを使う。先手（o）が勝ちの例を図1に示す。

o		x
o	x	
o	x	o

図1　先手（o）が勝ちの例

列挙型Markは記号を表し，先手はMark.CIRCLE，後手はMark.CROSSを使うこととする。また，升に何も書かれていない状態をMark.BLANKで表す。

クラスTicTacToeBoardは盤を表し，(1)～(4)のメソッドをもつ。

(1) Progress put(int x, int y, Mark mark)

盤の位置(x, y)にmarkで表される記号を書く。盤の状態を列挙型Progressで返す。戻り値と盤の状態の対応は，次のとおりである。

Progress.CIRCLE_WON — 先手の勝ち

Progress.CROSS_WON — 後手の勝ち

Progress.DRAWN — 引き分け

Progress.IN_PROGRESS — 進行中

指定位置が盤の範囲外のときはArrayIndexOutOfBoundsExceptionを投げる。既に勝敗が決まっているとき（引き分けを含む）はIllegalStateExceptionを投げ，次のいずれかに該当するときはIllegalArgumentExceptionを投げる。

① 指定位置に既に記号が書かれている。

② 指定された記号が書けるプレーヤの番ではない。

(2) `boolean check(int x, int y, int dx, int dy, Mark mark)`

盤のある一つの列の三つの升すべてに，指定された記号が書かれているか否かを検査する。位置`(x, y)`，`(x + dx, y + dy)`，`(x + 2 * dx, y + 2 * dy)`の三つの升に書かれている記号が，`mark`と等しければ`true`を返す。

(3) `Mark get(int x, int y)`

盤の位置`(x, y)`に書かれている記号を返す。

(4) `void undo()`

直前に書いた記号を消す。ただし，勝敗が決まった後は消すことができない。

クラス`TicTester`はテスト用のプログラムである。実行結果の一部を図2に示す。

```
Turn : CROSS : IN_PROGRESS
o   x
  x
o x o
Turn : CIRCLE : IN_PROGRESS
o   x
  x o
o x o
undo
o   x
  x
o x o
Turn : CIRCLE : CIRCLE_WON
o   x
o x
o x o
Game is over.
```

図2　実行結果の一部

〔参考：列挙（enum）について〕

プログラム中の`Mark`と`Progress`はそれぞれ列挙型であり，列挙型`Mark`は三つの列挙定数`CIRCLE`，`CROSS`，`BLANK`を，列挙型`Progress`は四つの列挙定数`CIRCLE_WON`，`CROSS_WON`，`DRAWN`，`IN_PROGRESS`をもつ。列挙型はクラスの一種であり，列挙定数は列挙型のインスタンスである。

〔プログラム1〕

```
public enum Mark {
   // 記号の種類を表す列挙。各列挙定数では定数 symbol を定義する。
   // 定数 symbol の値は各コンストラクタで与えられた値('o'など)である。
   CIRCLE('o'), CROSS('x'), BLANK(' ');
   public final char symbol;
   Mark(char symbol) {
      this.symbol = symbol;
   }
}
```

〔プログラム2〕

```
public class TicTacToeBoard {
   // ゲームの進行状況を表す列挙
   public enum Progress {CIRCLE_WON, CROSS_WON, DRAWN, IN_PROGRESS}
   private Mark[][] board = new Mark[3][3]; // 盤
   private Mark turn = Mark.CIRCLE;
   private Progress progress = Progress.IN_PROGRESS;
   private int count = 0;       // 盤に書かれた記号の個数
   private int lastx, lasty;    // 最後に記号が書かれた位置を保持する。

   public TicTacToeBoard() {    // 盤を初期化する。
      for (int i = 0; i < 3; i++)
         for (int j = 0; j < 3; j++)
            board[i][j] = [ a ];
   }

   public synchronized Progress put(int x, int y, Mark mark) {
      if ( [ b ] )
         throw new IllegalStateException();
      if (board[x][y] != Mark.BLANK || mark != turn)
         throw new IllegalArgumentException();
      board[x][y] = mark;
      lastx = x;
      lasty = y;
      count++;
      // 勝ちか否かを検査する。
      boolean game =
         // たてと横の列を検査する。
         check(x, 0, 0, 1, mark) || check(0, y, 1, 0, mark) ||
         // 斜めの列を検査する。
         check(0, 0, 1, 1, mark) || check(0, 2, [ c ], mark);
      if (game)
         if (turn == Mark.CIRCLE)
            progress = Progress.CIRCLE_WON;
         else progress = Progress.CROSS_WON;
      else if (count == 9) progress = Progress.DRAWN;
      // 次に書かれる記号の種類を決定する。
      if (turn == Mark.CIRCLE) turn = Mark.CROSS;
      else turn = Mark.CIRCLE;
      return progress;
   }
```

Java

```
    private boolean check(int x, int y, int dx, int dy, Mark mark) {
        return (board[x][y] == mark &&
                board[x + dx][y + dy] == mark &&
                board[x + 2 * dx][y + 2 * dy] == mark);
    }

    public Mark get(int x, int y) {
        return board[x][y];
    }

    public synchronized void undo() {
        if (progress == Progress.IN_PROGRESS &&
            board[lastx][lasty] != Mark.BLANK) {
            turn = board[lastx][lasty];
            board[lastx][lasty] = Mark.BLANK;
            count--;
        } else {
            throw new IllegalStateException();
        }
    }
}
```

〔プログラム3〕

```
public class TicTester {
    public static void main(String[] args) {
        // 記号を書く位置(x, y)を定義した配列。null のときは undo を呼ぶ。
        int[][] p = {{0, 0}, {1, 1}, {2, 2}, {2, 0}, {0, 2}, {1, 2},
                     {2, 1}, null, {0, 1}, {1, 0}};
        TicTacToeBoard board = new TicTacToeBoard();
        Mark marks[] = {Mark.CIRCLE, Mark.CROSS};
        for (int i = 0; i < p.length; i++) {
            try {
                if (p[i] == null) {
                    System.out.println("undo");
                    board.undo();
                } else {
                    Mark turn = marks[    d    ];
                    System.out.println("Turn : " + turn + " : " +
                        board.put(p[i][0], p[i][1], turn));
                }
                for (int y = 0; y < 3; y++) {
                    for (int x = 0; x < 3; x++)
                        System.out.print(board.get(x, y).symbol + " ");
                    System.out.println();
                }
            } catch (IllegalStateException ise) {
                System.out.println("Game is over.");
            }
        }
    }
}
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a に関する解答群

ア　`Mark.BLANK`　　イ　`Mark.CIRCLE`

ウ　`Mark.CROSS`　　エ　`null`

b に関する解答群

ア　`count < 0`

イ　`count >= 9`

ウ　`progress != Progress.IN_PROGRESS`

エ　`progress == Progress.IN_PROGRESS`

c に関する解答群

ア　`-1, -1`　　イ　`-1, 1`　　ウ　`1, -1`　　エ　`1, 1`

d に関する解答群

ア　`i`　　イ　`i % 2`　　ウ　`i & 2`　　エ　`i / 2`

設問2　クラス `TicTacToeBoard` のメソッド `undo` に関する記述として正しい答えを，解答群の中から二つ選べ。

解答群

ア　2手目以降で続けて2回呼ぶと，二つ消せる。

イ　2手目以降で続けて2回呼ぶと，呼ばなかった状態に戻る。

ウ　2手目以降で続けて2回呼んでも，消せるのは一つだけである。

エ　ゲーム中で先手後手とも，それぞれ1回しか消せない。

オ　消すことができなければ例外を投げる。

問 13　次のアセンブラプログラムの説明及びプログラムを読んで，設問 1，2 に答えよ。

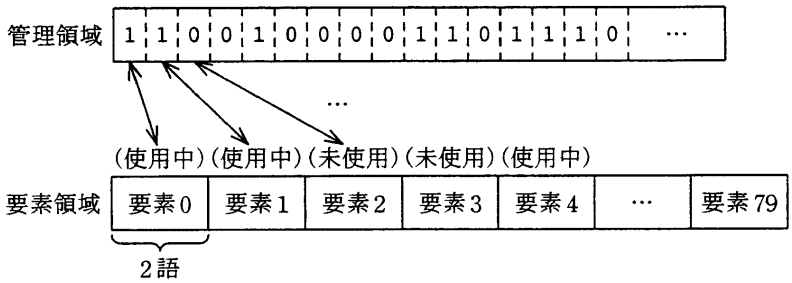
〔プログラムの説明〕

メモリプールの管理を行う副プログラム MPMGR である。

MPMGR は，割当て要求に対して，要素領域中の未使用の 1 要素を割り当てる。また，返却要求に対して，指定された 1 要素を未使用の状態にする。

(1) メモリプールは，要素領域と，それを管理する管理領域からなる。要素領域は連続する 80 要素からなり，各要素は連続する 2 語からなる。

各要素には，管理領域の 1 ビットずつが先頭から順に対応する。管理領域のビットの値が 1 のとき，対応する要素領域の要素は使用中とし，管理領域のビットの値が 0 のとき，対応する要素領域の要素は未使用とする。



(2) MPMGR は，GR1 に設定された値によって，次の処理を行う。括弧内は各処理に対応したラベルである。

0：初期化（MINI）

すべての要素を未使用に設定する。

1：要素の割当て（MSER）

未使用要素を検索し，見つかればその要素の先頭アドレスを GR0 に設定するとともに，その要素を使用中に設定する。未使用要素がなければ，GR0 に −1 を設定する。

2：要素の返却（MREL）

GR0 に設定されたアドレスに対応する要素を未使用に設定する。

(3) 副プログラムから戻るとき，汎用レジスタ GR1 ～ GR7 の内容は元に戻す。

〔プログラム〕

```
MPMGR   START
        RPUSH
        LD      GR1,LTBL,GR1
        [      a      ]      ; 指定された処理への分岐
; 初期化
MINI    LAD     GR3,0
        LD      GR0,GR3
LP      ST      GR0,MFRG,GR3
        LAD     GR3,1,GR3
        CPA     GR3,MFSZ
        JNZ     LP
        JUMP    FIN
; 要素の割当て
MSER    LAD     GR2,0
        LD      GR0,=-1         ; 戻り値の初期設定
LP1     LD      GR7,MFRG,GR2    ; 管理領域から１語取出し
        LAD     GR3,-1          ; １語内ビット位置カウンタ初期化
LP2     LAD     GR3,1,GR3
        CPA     GR3,=16         ; １語分のチェック終了か？
        JZE     CONT
        SLL     GR7,1
        [      b      ]
        JUMP    FIND
CONT    LAD     GR2,1,GR2
        CPA     GR2,MFSZ        ; 管理領域全体のチェック完了か？
        JZE     FIN
        JUMP    LP1
FIND    LD      GR1,=#8000      ; マスクパターンの生成
        SRL     GR1,0,GR3
        LD      GR7,MFRG,GR2    ; 対象ビットを含む語の取出し
        [      c      ]
        ST      GR7,MFRG,GR2
        SLL     GR2,4
        ADDL    GR2,GR3         ; 管理領域先頭から通しのビット位置
        SLL     GR2,1           ; 要素領域先頭からの差分
        LAD     GR0,MPDT,GR2
        JUMP    FIN
```

```
; 要素の返却
MREL    LD      GR2,GR0          ; GR2 ← 返却対象アドレス
        LAD     GR7,MPDT
        SUBL    GR2,GR7          ; 要素領域先頭からの差分
        SRL     GR2,1            ; 先頭から通しの要素位置
        LD      GR4,GR2          ; 管理領域先頭から通しのビット位置
        AND     GR4,=#000F       ; 1語内でのビット位置
        LD      GR1,=#8000       ; マスクパターンの生成
        SRL     GR1,0,GR4        ; 例:GR4=2の場合, GR1 ← 0010000000000000
                [      d      ]
        SRL     GR2,4            ; 対象ビットを含む語を指すための差分
        LD      GR7,MFRG,GR2
        AND     GR7,GR1          ; 対象ビットを未使用に設定
        ST      GR7,MFRG,GR2
FIN     RPOP
        RET
MFSZ    DC      5                ; 管理領域の大きさ
MFRG    DS      5                ; 管理領域
MPDT    DS      160              ; 要素領域
LTBL    DC      MINI             ; 分岐ラベル表
        DC      MSER
        DC      MREL
        END
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a に関する解答群

```
ア  CALL  0,GR1        イ  CALL  1,GR1        ウ  CALL  LTBL,GR1
エ  JUMP  0,GR1        オ  JUMP  1,GR1        カ  JUMP  LTBL,GR1
```

b に関する解答群

```
ア  JMI   LP1          イ  JMI   LP2          ウ  JNZ   LP1
エ  JNZ   LP2          オ  JOV   LP1          カ  JOV   LP2
```

c に関する解答群

```
ア  AND   GR7,GR1      イ  OR    GR7,GR1      ウ  SLL   GR7,0,GR1
エ  SLL   GR7,0,GR3    オ  SRL   GR7,0,GR1    カ  SRL   GR7,0,GR3
```

d に関する解答群

```
ア  AND   GR1,=#8000   イ  AND   GR1,=#FFFF   ウ  OR    GR1,=#8000
エ  OR    GR1,=#FFFF   オ  XOR   GR1,=#8000   カ  XOR   GR1,=#FFFF
```

アセンブラ

設問2　管理領域の最初の語の値が #C800 のとき，GR1 に 1 を設定して MPMGR を呼び出した。

ラベル FIND に制御が移ったときの GR3 の内容として，正しい答えを解答群の中から選べ。

解答群

ア　0	イ　1	ウ　2
エ　3	オ　4	カ　5

■ Javaプログラムで使用するAPIの説明

java.util

public interface Map<K, V>

型Kのキーに型Vの値を対応付けて保持するインタフェースを提供する。各キーは，一つの値としか対応付けられない。

メソッド

`public V get(Object key)`

指定されたキーに対応付けられた値を返す。

引数： key — キー

戻り値：指定されたキーに対応付けられた型Vの値
このキーと値の対応付けがなければ null

`public V put(K key, V value)`

指定されたキーに指定された値を対応付けて登録する。このキーが既にほかの値と対応付けられていれば，その値は指定された値に置き換えられる。

引数： key — キー
value — 値

戻り値：指定されたキーに対応付けられていた型Vの古い値
このキーと値の対応付けがなければ null

java.util

public class HashMap<K, V>

インタフェースMapのハッシュを用いた実装である。キー及び値は，nullでもよい。

コンストラクタ

`public HashMap()`

空のHashMapを作る。

メソッド

`public V get(Object key)`

インタフェースMapのメソッドgetと同じ

`public V put(K key, V value)`

インタフェースMapのメソッドputと同じ

java.lang

public final class String

クラス String は文字列を表す。

メソッド

---

public int compareTo(String anotherString)

二つの文字列を辞書順に比較した結果を整数値で返す。
二つの文字列の比較方法は〔参考〕に示す。
引数： anotherString ― 比較の対象となる文字列
戻り値：この文字列が引数文字列に等しいときは 0
この文字列が引数文字列より辞書順で小さいときは負の値
この文字列が引数文字列より辞書順で大きいときは正の値

〔参考〕

二つの文字列の最初の文字から順に比較していき，最初に異なる文字が現れた文字位置が k であるとき，次式の値をメソッドの戻り値とする。

```
this.charAt(k) - anotherString.charAt(k)
```

二つの文字列において異なる文字が現れず，かつ二つの文字列の長さが異なるとき，次式の値をメソッドの戻り値とする。

```
this.length() - anotherString.length()
```

二つの文字列において異なる文字が現れず，かつ二つの文字列の長さが一致したときは 0 を返す。

java.util

public final class Locale

クラス Locale は言語，国などを識別する。

フィールド

---

public static final Locale ENGLISH

英語を表す定数。

java.util

public class Arrays

クラス Arrays は，配列を操作するクラスメソッドからなる。

メソッド

`public static <T> void sort(T[] a, Comparator<? super T> c)`

指定された Comparator が規定する順序に従って，指定されたオブジェクトの配列要素を並べ替える。 Comparator が規定する順序が同じ要素同士の順番は，並べ替えによって変更されない。

引数： a — 並べ替えを行う配列
c — 配列要素の順序付けをする Comparator のオブジェクト

`public static String toString(Object[] a)`

指定されたオブジェクトの配列の文字列表現を返す。

引数： a — 配列

戻り値：a を表現する文字列

java.util

public interface Comparator<T>

あるオブジェクトの集合に対して完全な順序を規定する関数を提供するインタフェースである。

メソッド

`public int compare(T o1, T o2)`

引数で与えられた型 T の二つのオブジェクトを比較し，大小関係を整数値で返す。

引数： o1 — 1 番目のオブジェクト
o2 — 2 番目のオブジェクト

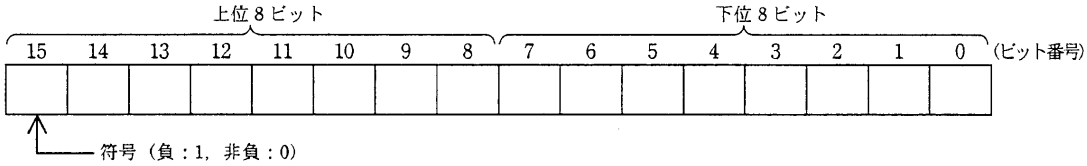
戻り値：o1 が o2 より小さいときは負の値
o1 と o2 が等しいときは 0
o1 が o2 より大きいときは正の値

■アセンブラ言語の仕様

1. システム COMETⅡの仕様

1.1 ハードウェアの仕様

(1) 1語は16ビットで，そのビット構成は，次のとおりである。



(2) 主記憶の容量は65536語で，そのアドレスは0～65535番地である。

(3) 数値は，16ビットの2進数で表現する。負数は，2の補数で表現する。

(4) 制御方式は逐次制御で，命令語は1語長又は2語長である。

(5) レジスタとして，GR（16ビット），SP（16ビット），PR（16ビット），FR（3ビット）の4種類がある。

GR（汎用レジスタ，General Register）は，GR0～GR7の8個があり，算術，論理，比較，シフトなどの演算に用いる。このうち，GR1～GR7のレジスタは，指標レジスタ（index register）としてアドレスの修飾にも用いる。

SP（スタックポインタ，Stack Pointer）は，スタックの最上段のアドレスを保持している。

PR（プログラムレジスタ，Program Register）は，次に実行すべき命令語の先頭アドレスを保持している。

FR（フラグレジスタ，Flag Register）は，OF（Overflow Flag），SF（Sign Flag），ZF（Zero Flag）と呼ぶ3個のビットからなり，演算命令などの実行によって次の値が設定される。これらの値は，条件付き分岐命令で参照される。

OF	算術演算命令の場合は，演算結果が−32768～32767に収まらなくなったとき1になり，それ以外のとき0になる。論理演算命令の場合は，演算結果が0～65535に収まらなくなったとき1になり，それ以外のとき0になる。
SF	演算結果の符号が負（ビット番号15が1）のとき1，それ以外のとき0になる。
ZF	演算結果が零（全部のビットが0）のとき1，それ以外のとき0になる。

(6) 論理加算又は論理減算は，被演算データを符号のない数値とみなして，加算又は減算する。

1.2 命令

命令の形式及びその機能を示す。ここで，一つの命令コードに対し2種類のオペランドがある場合，上段はレジスタ間の命令，下段はレジスタと主記憶間の命令を表す。

命　令	書き方		命令の説明	FRの設定
	命令コード	オペランド		

(1) ロード，ストア，ロードアドレス命令

ロード LoaD	LD	r1,r2	r1 ← (r2)	○*1
		r,adr [,x]	r ← (実効アドレス)	
ストア STore	ST	r,adr [,x]	実効アドレス ← (r)	−
ロードアドレス Load ADdress	LAD	r,adr [,x]	r ← 実効アドレス	

(2) 算術，論理演算命令

算術加算 ADD Arithmetic	ADDA	r1,r2	r1 ← (r1) + (r2)	○
		r,adr [,x]	r ← (r) + (実効アドレス)	
論理加算 ADD Logical	ADDL	r1,r2	r1 ← (r1) $+_L$ (r2)	
		r,adr [,x]	r ← (r) $+_L$ (実効アドレス)	
算術減算 SUBtract Arithmetic	SUBA	r1,r2	r1 ← (r1) − (r2)	
		r,adr [,x]	r ← (r) − (実効アドレス)	
論理減算 SUBtract Logical	SUBL	r1,r2	r1 ← (r1) $-_L$ (r2)	
		r,adr [,x]	r ← (r) $-_L$ (実効アドレス)	
論理積 AND	AND	r1,r2	r1 ← (r1) AND (r2)	○*1
		r,adr [,x]	r ← (r) AND (実効アドレス)	
論理和 OR	OR	r1,r2	r1 ← (r1) OR (r2)	
		r,adr [,x]	r ← (r) OR (実効アドレス)	
排他的論理和 eXclusive OR	XOR	r1,r2	r1 ← (r1) XOR (r2)	
		r,adr [,x]	r ← (r) XOR (実効アドレス)	

(3) 比較演算命令

算術比較 ComPare Arithmetic	CPA	r1,r2	(r1) と (r2)，又は (r) と (実効アドレス) の算術比較又は論理比較を行い，比較結果によって，FR に次の値を設定する。	○*1
		r,adr [,x]		
論理比較 ComPare Logical	CPL	r1,r2		
		r,adr [,x]		

比較結果	FR の値 SF	FR の値 ZF
(r1) ＞ (r2)	0	0
(r) ＞ (実効アドレス)		
(r1) ＝ (r2)	0	1
(r) ＝ (実効アドレス)		
(r1) ＜ (r2)	1	0
(r) ＜ (実効アドレス)		

(4) シフト演算命令

算術左シフト Shift Left Arithmetic	SLA	r,adr [,x]	符号を除き (r) を実効アドレスで指定したビット数だけ左又は右にシフトする。 シフトの結果，空いたビット位置には，左シフトのときは 0，右シフトのときは符号と同じものが入る。	○*2
算術右シフト Shift Right Arithmetic	SRA	r,adr [,x]		
論理左シフト Shift Left Logical	SLL	r,adr [,x]	符号を含み (r) を実効アドレスで指定したビット数だけ左又は右にシフトする。 シフトの結果，空いたビット位置には 0 が入る。	
論理右シフト Shift Right Logical	SRL	r,adr [,x]		

(5) 分岐命令

正分岐 Jump on PLus	JPL	adr [,x]	FR の値によって，実効アドレスに分岐する。分岐しないときは，次の命令に進む。	—
負分岐 Jump on MInus	JMI	adr [,x]		
非零分岐 Jump on Non Zero	JNZ	adr [,x]		
零分岐 Jump on ZEro	JZE	adr [,x]		
オーバフロー分岐 Jump on OVerflow	JOV	adr [,x]		
無条件分岐 unconditional JUMP	JUMP	adr [,x]	無条件に実効アドレスに分岐する。	

命令	分岐するときの FR の値 OF	SF	ZF
JPL		0	0
JMI		1	
JNZ			0
JZE			1
JOV	1		

(6) スタック操作命令

プッシュ PUSH	PUSH adr [,x]	SP ← (SP) $-_L$ 1, (SP) ← 実効アドレス	—
ポップ POP	POP r	r ← ((SP)), SP ← (SP) $+_L$ 1	

(7) コール，リターン命令

コール CALL subroutine	CALL adr [,x]	SP ← (SP) $-_L$ 1, (SP) ← (PR), PR ← 実効アドレス	—
リターン RETurn from subroutine	RET	PR ← ((SP)), SP ← (SP) $+_L$ 1	

(8) その他

スーパバイザコール SuperVisor Call	SVC adr [,x]	実効アドレスを引数として割出しを行う。実行後の GR と FR は不定となる。	—
ノーオペレーション No OPeration	NOP	何もしない。	

(注) r, r1, r2　いずれも GR を示す。指定できる GR は GR0 ～ GR7
adr　アドレスを示す。指定できる値の範囲は 0 ～ 65535
x　指標レジスタとして用いる GR を示す。指定できる GR は GR1 ～ GR7
[　]　[　] 内の指定は省略できることを示す。
(　)　(　) 内のレジスタ又はアドレスに格納されている内容を示す。
実効アドレス　adr と x の内容との論理加算値又はその値が示す番地
←　演算結果を，左辺のレジスタ又はアドレスに格納することを示す。
$+_L$, $-_L$　論理加算，論理減算を示す。
FR の設定　○　：設定されることを示す。
○*1：設定されることを示す。ただし，OF には 0 が設定される。
○*2：設定されることを示す。ただし，OF にはレジスタから最後に送り出されたビットの値が設定される。
－　：実行前の値が保持されることを示す。

1.3 文字の符号表

(1) JIS X 0201 ラテン文字・片仮名用 8 ビット符号で規定する文字の符号表を使用する。
(2) 右に符号表の一部を示す。1 文字は 8 ビットからなり，上位 4 ビットを列で，下位 4 ビットを行で示す。例えば，間隔，4，H，¥ のビット構成は，16 進表示で，それぞれ 20，34，48，5C である。16 進表示で，ビット構成が 21 ～ 7E（及び表では省略している A1 ～ DF）に対応する文字を図形文字という。図形文字は，表示（印刷）装置で，文字として表示（印字）できる。
(3) この表にない文字とそのビット構成が必要な場合は，問題中で与える。

行＼列	02	03	04	05	06	07
0	間隔	0	@	P	`	p
1	!	1	A	Q	a	q
2	"	2	B	R	b	r
3	#	3	C	S	c	s
4	$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	'	7	G	W	g	w
8	(	8	H	X	h	x
9	)	9	I	Y	i	y
10	*	:	J	Z	j	z
11	+	;	K	[	k	{
12	,	<	L	¥	l	|
13	-	=	M	]	m	}
14	.	>	N	^	n	~
15	/	?	O	_	o	

2. アセンブラ言語 CASLⅡの仕様

2.1 言語の仕様

(1) CASLⅡは，COMETⅡのためのアセンブラ言語である。
(2) プログラムは，命令行及び注釈行からなる。
(3) 1命令は1命令行で記述し，次の行へ継続できない。
(4) 命令行及び注釈行は，次に示す記述の形式で，行の1文字目から記述する。

行の種類		記述の形式
命令行	オペランドあり	[ラベル] {空白} {命令コード} {空白} {オペランド} [{空白} [コメント]]
	オペランドなし	[ラベル] {空白} {命令コード} [{空白} [{;} [コメント]]]
注釈行		[空白] {;} [コメント]

(注) [ ] [ ] 内の指定が省略できることを示す。
{ } { } 内の指定が必須であることを示す。
ラベル その命令の（先頭の語の）アドレスを他の命令やプログラムから参照するための名前である。長さは 1～8 文字で，先頭の文字は英大文字でなければならない。以降の文字は，英大文字又は数字のいずれでもよい。なお，予約語である GR0～GR7 は，使用できない。
空白 1文字以上の間隔文字の列である。
命令コード 命令ごとに記述の形式が定義されている。
オペランド 命令ごとに記述の形式が定義されている。
コメント 覚え書きなどの任意の情報であり，処理系で許す任意の文字を書くことができる。

2.2 命令の種類

命令は，4種類のアセンブラ命令（START，END，DS，DC），4種類のマクロ命令（IN，OUT，RPUSH，RPOP）及び機械語命令（COMETⅡの命令）からなる。その仕様を次に示す。

命令の種類	ラベル	命令コード	オペランド	機能
アセンブラ命令	ラベル	START	[実行開始番地]	プログラムの先頭を定義 プログラムの実行開始番地を定義 他のプログラムで参照する入口名を定義
		END		プログラムの終わりを明示
	[ラベル]	DS	語数	領域を確保
	[ラベル]	DC	定数 [,定数] …	定数を定義
マクロ命令	[ラベル]	IN	入力領域,入力文字長領域	入力装置から文字データを入力
	[ラベル]	OUT	出力領域,出力文字長領域	出力装置へ文字データを出力
	[ラベル]	RPUSH		GR の内容をスタックに格納
	[ラベル]	RPOP		スタックの内容を GR に格納
機械語命令	[ラベル]	（「1.2 命令」を参照）		

2.3 アセンブラ命令

アセンブラ命令は，アセンブラの制御などを行う。

(1)

START	[実行開始番地]

START 命令は，プログラムの先頭を定義する。
実行開始番地は，そのプログラム内で定義されたラベルで指定する。指定がある場合はその番地から，省略した場合は START 命令の次の命令から，実行を開始する。
また，この命令につけられたラベルは，他のプログラムから入口名として参照できる。

(2) | END | |

END 命令は，プログラムの終わりを定義する。

(3) | DS | 語数 |

DS 命令は，指定した語数の領域を確保する。

語数は，10 進定数（≧0）で指定する。語数を 0 とした場合，領域は確保しないが，ラベルは有効である。

(4) | DC | 定数［,定数］… |

DC 命令は，定数で指定したデータを（連続する）語に格納する。

定数には，10 進定数，16 進定数，文字定数，アドレス定数の 4 種類がある。

定数の種類	書き方	命　令　の　説　明
10 進定数	n	n で指定した 10 進数値を，1 語の 2 進数データとして格納する。ただし，n が−32768～32767 の範囲にないときは，その下位 16 ビットを格納する。
16 進定数	#h	h は 4 けたの 16 進数（16 進数字は 0～9，A～F）とする。h で指定した 16 進数値を 1 語の 2 進数データとして格納する（0000≦h≦FFFF）。
文字定数	'文字列'	文字列の文字数（＞0）分の連続する領域を確保し，最初の文字は第 1 語の下位 8 ビットに，2 番目の文字は第 2 語の下位 8 ビットに，…と順次文字データとして格納する。各語の上位 8 ビットには 0 のビットが入る。 文字列には，間隔及び任意の図形文字を書くことができる。ただし，アポストロフィ（'）は 2 個続けて書く。
アドレス定数	ラベル	ラベルに対応するアドレスを 1 語の 2 進数データとして格納する。

2.4 マクロ命令

マクロ命令は，あらかじめ定義された命令群とオペランドの情報によって，目的の機能を果たす命令群を生成する（語数は不定）。

(1) | IN | 入力領域,入力文字長領域 |

IN 命令は，あらかじめ割り当てた入力装置から，1 レコードの文字データを読み込む。

入力領域は，256 語長の作業域のラベルであり，この領域の先頭から，1 文字を 1 語に対応させて順次入力される。レコードの区切り符号（キーボード入力の復帰符号など）は，格納しない。格納の形式は，DC 命令の文字定数と同じである。入力データが 256 文字に満たない場合，入力領域の残りの部分は実行前のデータを保持する。入力データが 256 文字を超える場合，以降の文字は無視される。

入力文字長領域は，1 語長の領域のラベルであり，入力された文字の長さ（≧0）が 2 進数で格納される。ファイルの終わり（end of file）を検出した場合は，−1 が格納される。

IN 命令を実行すると，GR の内容は保存されるが，FR の内容は不定となる。

(2) | OUT | 出力領域,出力文字長領域 |

OUT 命令は，あらかじめ割り当てた出力装置に，文字データを，1 レコードとして書き出す。

出力領域は，出力しようとするデータが 1 文字 1 語で格納されている領域のラベルである。格納の形式は，DC 命令の文字定数と同じであるが，上位 8 ビットは，OS が無視するので 0 でなくてもよい。

出力文字長領域は，1 語長の領域のラベルであり，出力しようとする文字の長さ（≧0）を 2 進数で格納しておく。

OUT 命令を実行すると，GR の内容は保存されるが，FR の内容は不定となる。

(3) | RPUSH |

RPUSH 命令は，GR の内容を，GR1，GR2，…，GR7 の順序でスタックに格納する。

(4) | RPOP |

RPOP 命令は，スタックの内容を順次取り出し，GR7，GR6，…，GR1 の順序で GR に格納する。

2.5 機械語命令

機械語命令のオペランドは，次の形式で記述する。

r, r1, r2	GR は，記号 GR0 ～ GR7 で指定する。
x	指標レジスタとして用いる GR は，記号 GR1 ～ GR7 で指定する。
adr	アドレスは，10 進定数，16 進定数，アドレス定数又はリテラルで指定する。 リテラルは，一つの 10 進定数，16 進定数又は文字定数の前に等号（=）を付けて記述する。CASL Ⅱ は，等号の後の定数をオペランドとする DC 命令を生成し，そのアドレスを adr の値とする。

2.6 その他

(1) アセンブラによって生成される命令語や領域の相対位置は，アセンブラ言語での記述順序とする。ただし，リテラルから生成される DC 命令は，END 命令の直前にまとめて配置される。

(2) 生成された命令語，領域は，主記憶上で連続した領域を占める。

3. プログラム実行の手引

3.1 OS

プログラムの実行に関して，次の取決めがある。

(1) アセンブラは，未定義ラベル（オペランド欄に記述されたラベルのうち，そのプログラム内で定義されていないラベル）を，他のプログラムの入口名（START 命令のラベル）と解釈する。この場合，アセンブラはアドレスの決定を保留し，その決定を OS に任せる。OS は，実行に先立って他のプログラムの入口名との連係処理を行いアドレスを決定する（プログラムの連係）。

(2) プログラムは，OS によって起動される。プログラムがロードされる主記憶の領域は不定とするが，プログラム中のラベルに対応するアドレス値は，OS によって実アドレスに補正されるものとする。

(3) プログラムの起動時に，OS はプログラム用に十分な容量のスタック領域を確保し，その最後のアドレスに 1 を加算した値を SP に設定する。

(4) OS は，CALL 命令でプログラムに制御を渡す。プログラムを終了し OS に制御を戻すときは，RET 命令を使用する。

(5) IN 命令に対応する入力装置，OUT 命令に対応する出力装置の割当ては，プログラムの実行に先立って利用者が行う。

(6) OS は，入出力装置や媒体による入出力手続の違いを吸収し，システムでの標準の形式及び手続（異常処理を含む）で入出力を行う。したがって，IN，OUT 命令では，入出力装置の違いを意識する必要はない。

3.2 未定義事項

プログラムの実行等に関し，この仕様で定義しない事項は，処理系によるものとする。

〔 メ モ 用 紙 〕

〔 メ モ 用 紙 〕

〔 メ モ 用 紙 〕

6. 途中で退室する場合には，手を挙げて監督員に合図し，答案用紙が回収されてから静かに退室してください。

退室可能時間	13:40 ～ 15:20

7. **問題に関する質問にはお答えできません。**文意どおり解釈してください。
8. 問題冊子の余白などは，適宜利用して構いません。
9. Java プログラムで使用する API の説明及びアセンブラ言語の仕様は，この冊子の末尾を参照してください。
10. 電卓は，使用できません。
11. 試験終了後，この問題冊子は持ち帰ることができます。
12. 答案用紙は，白紙であっても提出してください。
13. 試験時間中にトイレへ行きたくなったり，気分が悪くなったりした場合は，手を挙げて監督員に合図してください。